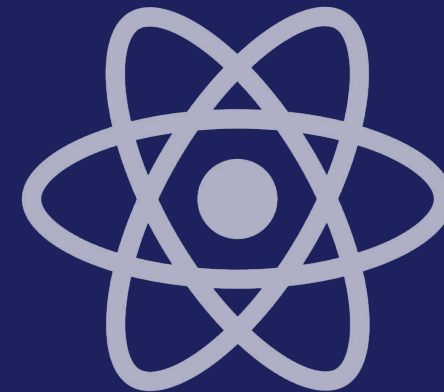


VENTURUS 4TECH



VENTURUS4TECH



Exercício

1. Vamos fazer os handlers para:
 - a. EDITAR
 - b. ADICIONAR
 - i. teremos um problema com a hierarquia dos componentes aqui (!)



















DESAFIO (LEVEL HARD):

- Validar o preenchimento obrigatório dos campos (ao clicar em submit): nome, descrição e salário - escreva o erro abaixo deles usando as classes do bootstrap:
 - *Bootstrap Form Validation*

```
<input type="text" class="form-control is-invalid" required>
<div class="invalid-feedback">
  Please choose a username.
</div>
```

Conexão com server

- Vamos usar uma biblioteca JS chamada **AXIOS**
 - Usa Promises
 - Fácil configuração

Browser Support					
					
Latest ✓	Latest ✓	Latest ✓	Latest ✓	Latest ✓	8+ ✓
 Firefox	 Chrome	 IE	 Edge	 Safari	
61  7 ✓	67  7 ✓	9  7 ✗	17  10 ✓	9  10.11 ✓	
		10  8 ✓			
		11  8.1 ✓			



Saiba mais: [Github do Axios](#)



Promises

- Assíncrona
- Permite tratamento do sucesso (then) ou erro (catch)
- Não trava a execução da aplicação

```
new Promise(executor)
```

```
state: "pending"  
result: undefined
```

resolve(value)

reject(error)

```
state: "fulfilled"  
result: value
```

```
state: "rejected"  
result: error
```



Saiba mais: [MDN Web Docs - Promise](#), [Promessas em JavaScript: uma introdução](#), [Javascript.info - Promise Basics](#)



Exercício

Vamos criar as conexões com o servidor:

```
$ npm install axios --save
```

1. Usar o axios
 - a. com GET para buscar todas as vagas
 - b. com POST para cadastrar nova vaga
 - i. vamos ter que fazer uma alteração no POST do servidor node
2. DESAFIO:
 - a. Faça a exclusão da vaga com o DELETE apenas se a confirmação do `window.confirm()` for verdadeira
 - i. Use o método de array `splice()` para remover o item em questão
 - b. Faça a alteração da vaga com PUT

CHALLENGE ACCEPTED



Desafio

Vamos criar um componente Loading para entender a chamada do método render()

Esse componente será mostrado na tela enquanto a lista de vagas não é exibida

- Crie um arquivo Loading.css também e importe junto ao componente:
 - `import './Loading.css';`
 - Use um estilo pronto de: icons8.com/cssload/
- Importe em JobsList e use enquanto não houver dados em `this.state.jobs`

A photograph of an empty classroom with rows of orange plastic desks and chairs. A large, dark brown chalkboard is mounted on the wall at the front of the room. The floor is made of light-colored square tiles.

react-router-dom

- Define as rotas da aplicação
- Não recarregam a página (SPA)
- É possível definir a parte da tela que vai ser atualizada e qual o componente a ser exibido



Saiba mais: [Paul Sherman - A Simple React Router v4 Tutorial](#)



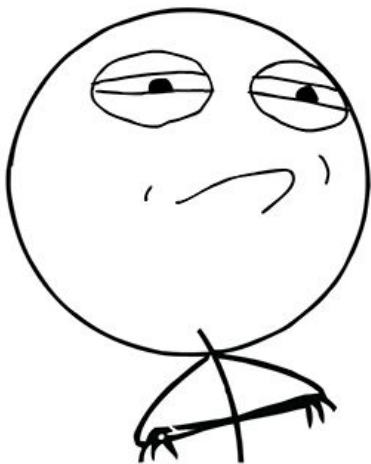
Exercício

Vamos criar as rotas da aplicação:

```
$ npm install react-router-dom --save
```

1. Usar o react-router-dom:
 - a. Vamos criar os links para o header da aplicação: Vagas e Sobre
 - b. Crie um componente novo chamado About e coloque qualquer HTML
 - c. Vamos usar agora os componentes pré-definidos do react-router-dom

CHALLENGE ACCEPTED



Desafio

- Crie um componente para exibir as informações completas de uma vaga ao clicar no card (ou um novo botão dentro do card)
- DICAS:
 - Vai ser necessário criar uma rota que receba um parâmetro: o id da vaga
 - Dê uma olhada nesse link: [Access Route Params in React Router v4](#)
 - Ao entrar no componente, você vai precisar fazer um GET no Axios passando o id da vaga



LOCALSTORAGE



Local Storage

- Permite armazenar informações no browser
- São mantidas por url
- Facilmente manipuladas
- Guardam informações simples



Saiba mais: [MDN - Window.localStorage](https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage)



Exercício

Criar mecanismo de autenticação na aplicação:

- Criar componente login
- Chamar validação do login pelo axios
- Armazenar token em **localStorage**
- Permitir visualização dos botões de ação apenas se o usuário estiver logado
- Enviar token nas chamadas de POST, PUT e DELETE

Para criar rotas “privadas”: [Protected routes and authentication with React Router v4](#)



+Links e referências

- Mais para aprender react:
 - [MDN Tutorials - WebHow to Learn React—A roadmap from beginner to advanced](#)
 - [Docs React.js - Main Concepts](#)
 - [Learning React.js: Getting Started and Concepts](#)
 - [Luiz Guerra - Sim, o React está tomando conta do desenvolvimento front-end. A questão é por quê?](#)

**Missão #DIA9
concluída!**

